Date=13/07/2020
Lecture By=Shubham Joshi
Notes By=Upadhyay Hemanshu
Subject ⇒ Python Lists

IN PREVIOUS LECTURE **(QUICK RECAP) Date-10/07/2020**	**In today's Lecture (Overview)**
⇒ **What is Def in Python??** ⇒ **What is Return In python??** ⇒ **Arrays In Python** ⇒ **What is List??** ⇒ **What is len??** ⇒ **Append In Python** ⇒ **Questions For Self Practice..**	=== We learned More Things About Lists/Types of Lists ⇒ **For Each loop In Python** ⇒ **Enumerate** In Python ⇒ **Comprehension and Slicing** ⇒ **Dictionary In Python** ⇒ **Questions For Self Practice..**

⇒ For Each Loop In Python

-**Python** For **Loops**. A for **loop** is **used for iterating over a sequence** (that is either a list, a tuple, a dictionary, a set, or a string).

Example

Print each fruit in a fruit list:

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  print(x)
```

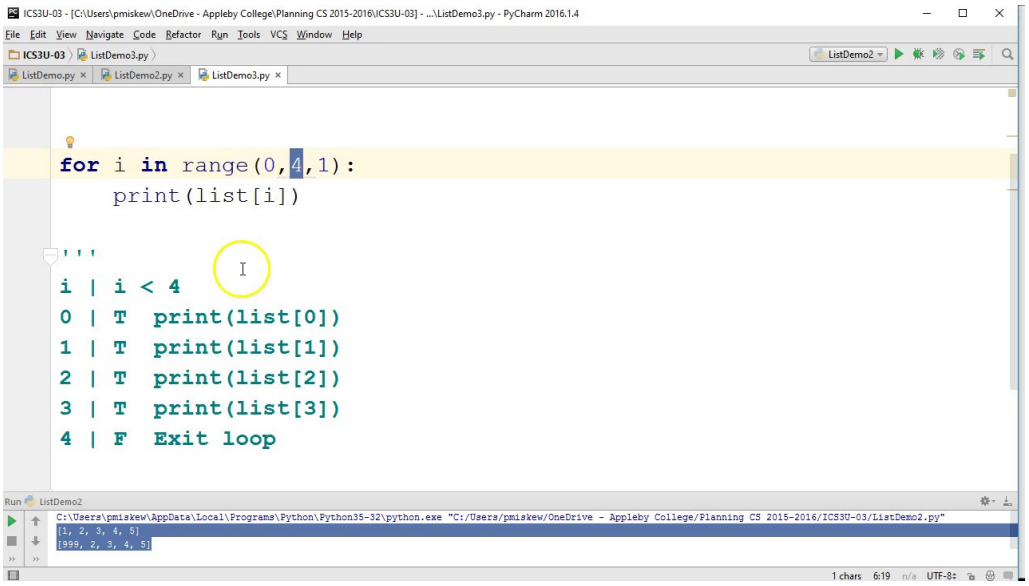
Looping Through a String

Even strings are iterable objects, they contain a sequence of characters:

Example

Loop through the letters in the word "banana":

```
for x in "banana":

  print(x)
```



loop through a set of code a specified number of times, we can use the `range()` function,

The `range()` function returns a sequence of numbers, starting from 0 by default, and increments by 1 (by default), and ends at a specified number.

Example

Using the range() function:

```
for x in range(6):

  print(x)
```

The `range()` function defaults to 0 as a starting value, however it is possible to specify the starting value by adding a parameter: `range(2, 6)`, which means values from 2 to 6 (but not including 6):

Example

Using the start parameter:

```
for x in range(2, 6):

print(x)
```

```
0 references
class Class
{
    0 references
    public void MyMethod()
    {
        List<string> greetings = new List<string>()
                { "hi", "yo", "hello", "howdy" };

        IEnumerable<string> enumerable()
        {
            foreach(string greet in greetings)
            {
                if(greet.Length < 3)
                {
                    yield return greet;
                }
            }

            yield break;
        }
```

"Click Here" To Know More About It

For Video Tutorial "Click Here"

⇒ Enumerate In Python

-A lot of times when dealing with iterators, we also get a **need to keep a count of iterations.** Python eases the programmers' task by providing a built-in function enumerate() for this task.

-The enumerate() function **takes a collection and returns it as an enumerate object.**

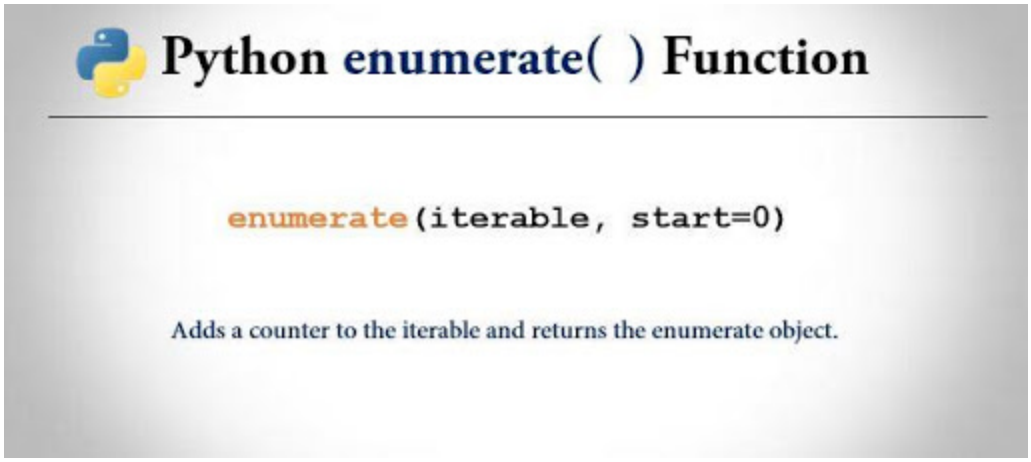
-The enumerate() function adds a counter as the key of the enumerate object.

Parameter	Description
iterable	An iterable object
start	A Number. Defining the start number of the enumerate object. Default 0

Example

Convert a tuple into an enumerate object:

```
x = ('apple', 'banana', 'cherry')
y = enumerate(x)
```




```
name_list=['john','justin','bob','petter']
for counter, value in enumerate(name_list):
    print(counter,value)
```

```
0 john
1 justin
2 bob
3 petter
```

"Click Here" To Know More About It

For Video Tutorial "Click Here"

⇒ Comprehension and Slicing

Comprehension

-Comprehensions **in Python provide us with a short and concise way to construct new sequences (such as lists, set, dictionary etc.)** using **sequences which have been already defined.**

Example

```
new_list = []
for i in old_list:
    if filter(i):
        new_list.append(expressions(i))
```

Create a simple list

Let's start easy by creating a simple list.

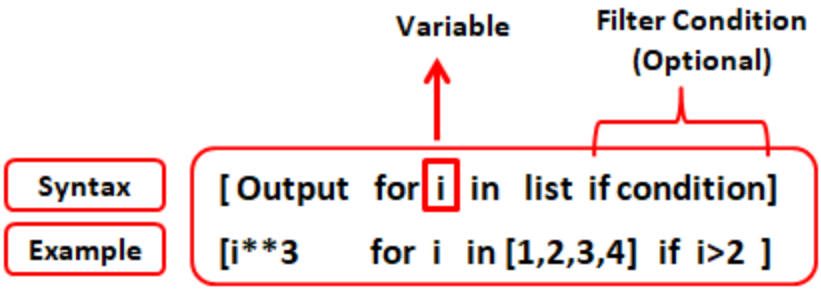
```
x = [i for i in range(10)]
print x

# This will give the output:
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Create a list using loops and list comprehension

For the next example, assume we want to create a list of squares. Start with an empty list.

```
# You can either use loops:
squares = []

for x in range(10):
```

```
    squares.append(x**2)

print squares

[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]

# Or you can use list comprehensions to get the same result:

squares = [x**2 for x in range(10)]

print squares

[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```



For Video Tutorial "Click Here"

=Slicing

-The **slice**() function **returns a slice object.**

- **slice** object is used to specify **how to slice a sequence.** You can specify where to start the **slicing**, and where to end.

-You can also specify the step, which allows you to e.g. **slice** only every other item.

Parameter Values

Parameter	Description
start	Optional. An integer number specifying at which position to start the slicing. Default is 0Optional. An integer number specifying at which position to start the slicing. Default is
end	An integer number specifying at which position to end the slicing
step	Optional. An integer number specifying the step of the slicing. Default is 1

Example

Create a slice object. Start the slice object at position 3, and slice to position 5, and return the result:

```
a = ("a", "b", "c", "d", "e", "f", "g", "h")
x = slice(3, 5)
print(a[x])
```

Example Get substring using slice object

```
# Program to get a substring from the given string

py_string = 'Python'

# stop = 3
```

```
# contains 0, 1 and 2 indices
slice_object = slice(3)
print(py_string[slice_object])  # Pyt

# start = 1, stop = 6, step = 2
# contains 1, 3 and 5 indices
slice_object = slice(1, 6, 2)
print(py_string[slice_object])   # yhn
```

Output

```
Pyt
yhn
```

Example Get substring using negative index

```
py_string = 'Python'

# start = -1, stop = -4, step = -1
# contains indices -1, -2 and -3
slice_object = slice(-1, -4, -1)
print(py_string[slice_object])   # noh
```

Output

```
noh
```

Example Using Indexing Syntax for Slicing

The slice object can be substituted with the indexing syntax in Python. You can alternately use the following syntax for slicing:

Obj[start:stop:step]

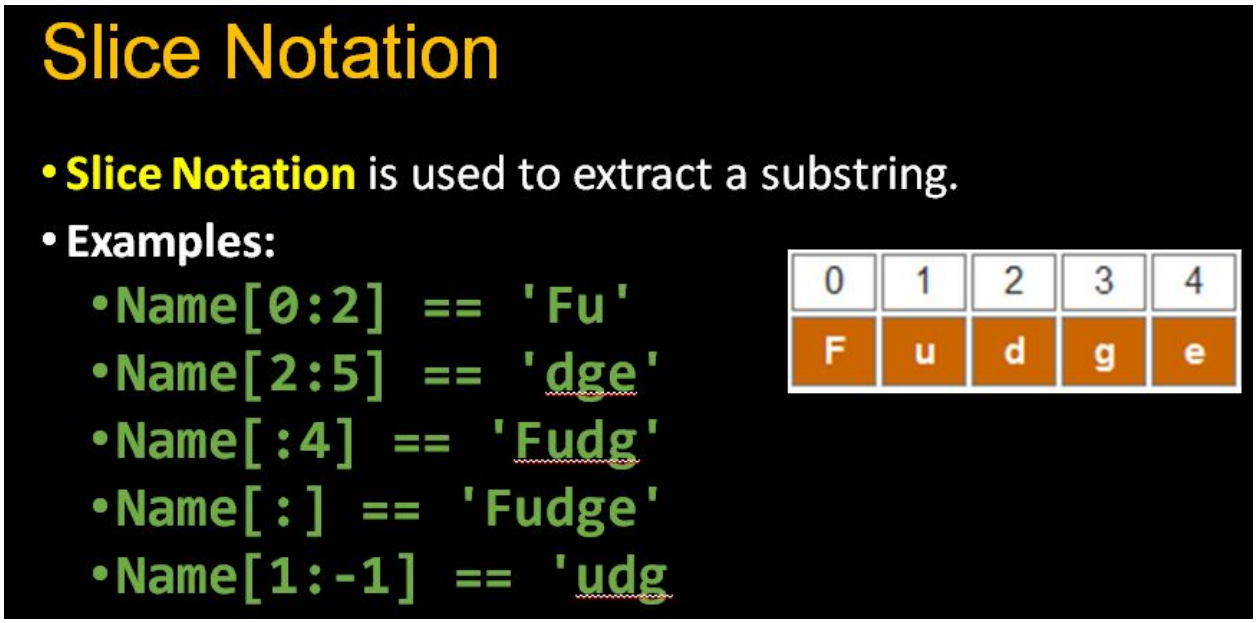
For example,

```
py_string = 'Python'

# contains indices 0, 1 and 2
```

```
print(py_string[0:3])  # Pyt

# contains indices 1 and 3
print(py_string[1:5:2]) # yh
```

Output

```
Pyt
yh
```




```
                  0    1    2    3    4    5    6
>>> letters = ['c', 'h', 'e', 'c', 'k', 'i', 'o']
                       2

>>> letters[1:4:2]
['h', 'c']
```

For Video Tutorial "Click Here"

⇒ Dictionary In Python

-A dictionary is a collection which is **unordered, changeable and indexed.** In Python dictionaries are written with curly brackets, and they have keys and values.

Example

Create and print a dictionary:

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
print(thisdict)
```

Check if Key Exists

To determine if a specified key is present in a dictionary use the `in` keyword:

Example

Check if "model" is present in the dictionary:

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
if "model" in thisdict:
  print("Yes, 'model' is one of the keys in the thisdict dictionary")
```

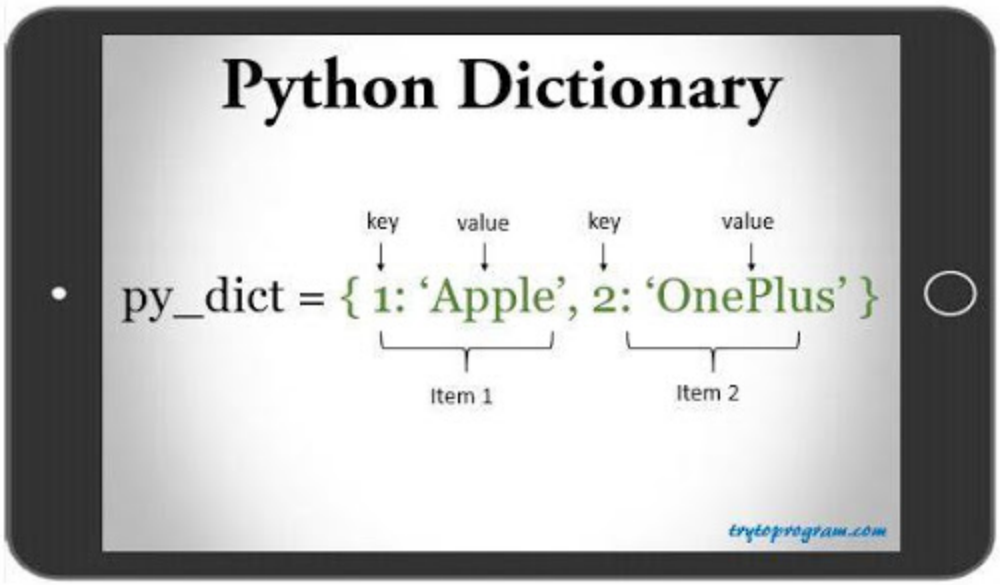
Dictionary Length

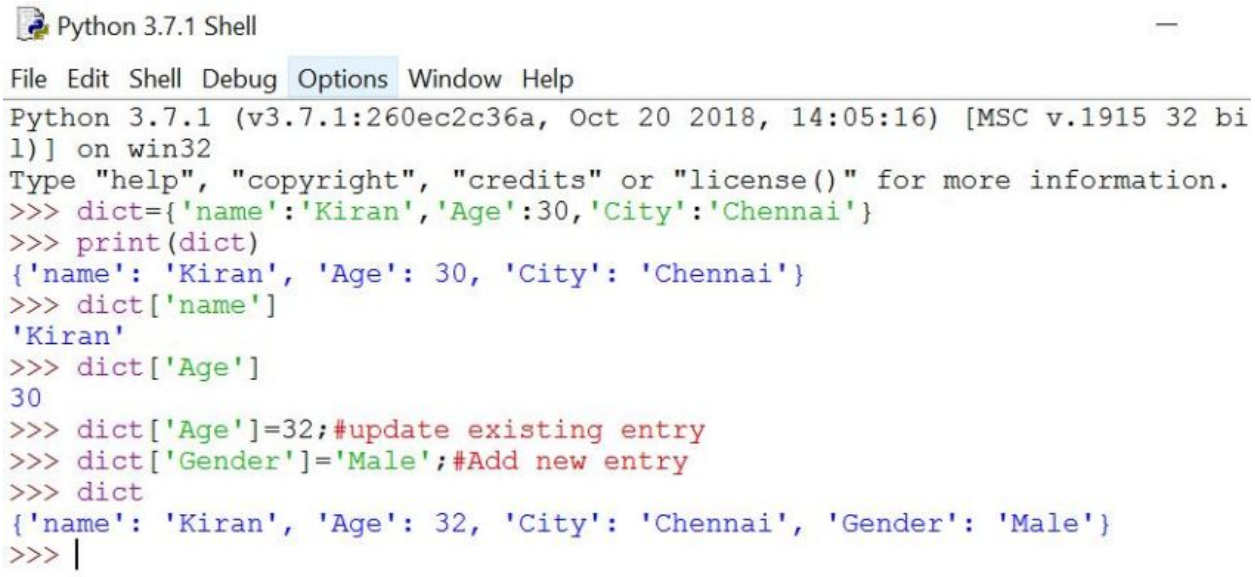
To determine how many items (key-value pairs) a dictionary has, use the `len()` function.

Example

Print the number of items in the dictionary:

```
print(len(thisdict))
```





"Click Here" To Know More About It

For Video Tutorial "Click Here"

⇒ Questions For Self Practice/CC For This Day

https://leetcode.com/problems/day-of-the-year/
https://leetcode.com/problems/add-strings/ (solve it without type casting)